Esquemas de firma digital con verificación distribuida

J. Herranz[1], A. Ruiz[2] y G. Sáez[2]

***Resumen*—En algunas ocasiones, una persona que ha firmado digitalmente un documento puede preferir que no todo el mundo tenga la capacidad para verificar la validez de la firma. Hay varios tipos de firma digital en la literatura que ofrecen diferentes tipos de restricciones a la hora de verificar.**

En este trabajo consideramos que el firmante ha elegido previamente un conjunto de verificadores y una familia de subconjuntos autorizados a verificar. De esta manera, una firma podrá ser verificada sólo si cooperan los verificadores de un subconjunto autorizado; si eso no ocurre, no se podrá obtener ninguna información sobre la validez o invalidez de la firma.

Después de describir formalmente las propiedades de seguridad que debe cumplir un esquema para esta funcionalidad, proponemos aquí un esquema concreto, basado en el esquema de firma de Schnorr, que mejora en eficiencia computacional la solución genérica consistente en combinar un esquema de firma digital con un esquema de cifrado con descifrado distribuido.

***Palabras clave*—Firma digital, estructuras de acceso, compartición de secretos, seguridad demostrable.**

I. Introducción

CADA vez es más frecuente realizar diferentes actividades (comerciales, profesionales, de ocio...) de manera digital. Una de las herramientas más importantes para asegurar el buen uso de los medios digitales son las *firmas digitales*, que proporcionan autenticidad e integridad a la información digital. Una tercera propiedad de una firma digital es que proporciona *no repudio*; es decir, un firmante real no puede negar en el futuro su asociación con el mensaje y afirmar que fue firmado por un tercero. Además, las implementaciones normales de un esquema de firma digital permiten la *verificación universal*: todo el mundo puede verificar la validez de una firma, ejecutando un protocolo de verificación que toma como entradas el mensaje, la firma y la clave pública del firmante.

En algunas situaciones, sin embargo, la propiedad de verificación universal puede ser demasiado fuerte, si el firmante desea un cierto nivel de anonimato o de privacidad. Por ejemplo, a la hora de apoyar una declaración pública de huelga, un trabajador de una empresa puede desear que los otros trabajadores de su nivel sean capaces de verificar su firma, pero no así el jefe de la empresa. O a la hora de filtrar un rumor (real) a la prensa rosa, el difundidor puede exigir la colaboración de un cierto subconjunto de periodistas para que la verificación tenga éxito y el rumor se convierta en noticia. Incluso el difundidor puede desear permanecer en el anonimato, dando sólo la información que pertenece a un cierto colectivo fiable, pero sin identificarse.

[1]IIIA-CSIC, Bellaterra, Spain. jherranz@iiia.csic.es.
[2]MAIV, UPC, Barcelona, Spain. {aruiz,german}@ma4.upc.edu.

Motivados por estas posibilidades, varios autores han propuesto diferentes extensiones del concepto básico de firma digital, con el fin de restringir de una manera u otra la capacidad de verificación e identificación. Por ejemplo, en las firmas *innegables* [3], la interacción con el firmante es necesaria a la hora de verificar una firma. De manera similar, en las firmas *con confirmador* [4], es necesario interactuar o bien con el firmante o bien con un cierto confirmador designado por el firmante, para poder verificar. En una firma *con verificador(es) designado(s)* [8], [9], sólo el/los usuario(s) designado(s) por el firmante acaba(n) convencido(s) de que la firma proviene realmente del firmante real. En una firma *de anillo* [13], el verificador queda convencido de que la firma proviene de alguno de los firmantes especificados en una lista (o anillo), pero no puede identificar al firmante real.

En este trabajo proponemos un enfoque diferente para restringir la verificación de una firma digital. Pensando en la responsabilidad por el documento firmado, el firmante prefiere restringir la capacidad de firma de dos maneras: primero, especificando un conjunto de verificadores; segundo, definiendo una familia (o *estructura de acceso*) de subconjuntos de verificadores autorizados a verificar. De esta manera, después de que el firmante calcula y publica una firma, se deben cumplir dos propiedades: (1) si los verificadores de un subconjunto autorizado colaboran, pueden verificar la validez de la firma y convencerse de que proviene del firmante, ya que el esquema es *infalsificable* y nadie diferente del firmante podría haber calculado una firma válida; (2) si los verificadores de un subconjunto no autorizado colaboran, no pueden obtener ninguna información acerca de si una firma es válida para un cierto mensaje o no. Llamaremos a este tipo de esquemas *firmas con verificación distribuida*. Aplicaciones para esta clase de esquemas de firma son, como hemos mencionado previamente, todas las situaciones donde la privacidad del firmante es importante, por ejemplo si el mensaje firmado contiene información personal delicada para el firmante: firma de documentos médicos, información sobre impuestos, transacciones y negocios personales, etc. Un ejemplo más concreto de aplicación podría ser el diseño de esquemas de firma conjunta de contratos de manera justa [5]: los firmantes sólo obtienen la firma final global si todos ellos han ejecutado correctamente el protocolo; si no es así, ninguno puede obtener firmas parciales de los demás, ni la firma global.

Hay algunos trabajos en la literatura que tratan este tipo de firmas, por ejemplo [10], [11], pero bajo otro enfoque. La primera diferencia es que estos trabajos sólo consideran estructuras de acceso de tipo *umbral* (i.e. los subconjuntos autorizados son aquellos que contienen un mínimo número

de participantes). La segunda diferencia es que en estos esquemas, el firmante escoge los verificadores y la estructura de acceso justo antes de firmar, y posiblemente estos conjuntos son diferentes para cada firma. Como consecuencia, en los esquemas de [10], [11], o bien la longitud de las firmas o bien el coste computacional requerido para firmar dependen linealmente del número total de verificadores. Contrariamente, nosotros consideramos en este trabajo la situación en que el firmante define desde el principio el conjunto de verificadores y la estructura de acceso. Este punto permite encontrar soluciones más eficientes, en las que la longitud de la firma y el coste computacional del protocolo de firma no dependen del número de verificadores.

Por ejemplo, podemos considerar la siguiente construcción genérica de un esquema de firma con verificación distribuida, a partir de un esquema estándar de firma digital Σ (usando por ejemplo [14]) y un esquema de cifrado con descifrado distribuido Ω (ver por ejemplo [15]). En la fase de inicialización, cada firmante escoge su estructura de acceso de verificadores, y reparte fragmentos de la clave privada de descifrado del esquema Ω entre los verificadores. Después, usa su clave privada de firma para firmar un mensaje, según el esquema Σ, y luego cifra la firma resultante con la clave pública del esquema de cifrado Ω. El texto cifrado resultante es la firma final. Si un conjunto autorizado de verificadores colabora, pueden ejecutar el protocolo de descifrado distribuido de Ω, usando sus fragmentos de la clave privada, y de esta manera descifrar la firma recibida para obtener una firma estándar según Σ, cuya validez puede entonces ser verificada usando la clave pública del firmante. Se puede demostrar que si tanto el esquema de firma Σ como el esquema de cifrado Ω alcanzan los niveles máximos de seguridad, entonces el esquema de firma digital con verificación distribuida resultante también alcanza las propiedades deseadas de seguridad. En cuanto a la eficiencia, la longitud de la firma y el coste computacional para firmar no dependen del número de verificadores. Sin embargo, el hecho de tener que firmar y cifrar ya supone un cierto coste computacional, que nunca será inferior a dos exponenciaciones modulares, por ejemplo.

Nuestro objetivo en este artículo es proponer algún esquema particular de firma digital con verificación distribuida que sea computacionalmente más eficiente que los esquemas que se derivan de esta construcción genérica. Nuestro esquema está basado en el esquema de firma de Schnorr, óbviamente con algunas modificaciones necesarias para permitir la propiedad de la verificación distribuida. Al igual que en la construcción genérica, la longitud de las firmas y el coste computacional del protocolo de firma son independientes del número total de verificadores. Además, la única operación costosa en el protocolo de firma es una exponenciación modular, lo que mejora la eficiencia de la construcción genérica. En cuanto a la seguridad, demostramos formalmente la infalsificabilidad del esquema, y también demostramos que tiene la propiedad de privacidad débil. Lamentablemente, no hemos sido capaces hasta ahora de demostrar que nuestro esquema disfruta del máximo nivel de privacidad, que denotamos aquí como privacidad fuerte.

El resto del artículo está organizado de la siguiente manera. En la Sección II, describimos los protocolos que forman parte en un esquema de firma con verificación distribuida. Las propiedades de seguridad requeridas para este tipo de esquemas están formalmente detalladas en la Sección III. Proponemos nuestro esquema particular en la Sección IV, y analizamos su seguridad en la Sección V. Por último, concluimos el trabajo en la Sección VI, donde también explicamos algunos problemas que quedan abiertos a futuros trabajos de investigación.

II. Esquemas de firma con verificación distribuida

Cada participante i en el sistema tendrá un par de claves secreta y pública (sk_i, pk_i). Un participante utilizará su clave privada sk_i para firmar. Para que la verificación de la firma resultante pueda ser distribuida, cierta información debe ser repartida entre un conjunto de n participantes $\mathcal{P} = \{1, \ldots, n\}$. Se define para cada $i \in \mathcal{P}$ una familia monótona creciente de subconjuntos autorizados, $\Gamma^{(i)} \subset 2^{\mathcal{P}-\{i\}}$ (es decir, que verifica: si $A_1 \in \Gamma^{(i)}$ y $A_1 \subset A_2 \subset \mathcal{P} - \{i\}$, entonces $A_2 \in \Gamma^{(i)}$), llamada *estructura de acceso* y determinada por su familia de subconjuntos minimales (o *base*) $\Gamma_0^{(i)} = \{A \in \Gamma^{(i)} \mid A - \{j\} \notin \Gamma^{(i)}, \text{ para todo } j \in A\}$. Esta estructura de acceso $\Gamma^{(i)}$ determina la familia de subconjuntos de participantes autorizados a verificar las firmas del participante i. En otras palabras, los participantes de un subconjunto autorizado $A \in \Gamma^{(i)}$ tienen que cooperar para poder verificar la firma de un mensaje m proveniente del usuario i. Por otro lado, una coalición de participantes fuera de $\Gamma^{(i)}$ no obtendrá ninguna información sobre la validez (o invalidez) de una firma producida por el usuario i. En nuestro modelo, suponemos que un usuario puede ser firmante o verificador. Por tanto, el conjunto $\mathcal{P} = \{1, \ldots, n\}$ contiene a todos los protagonistas del sistema.

Un $(\mathcal{P}, \Gamma^{(1)}, \ldots, \Gamma^{(n)})$-esquema de firma con verificación distribuida consiste en cuatro protocolos probabilísticos y en tiempo polinómico:

1) *Ini*: Este protocolo puede ser ejecutado por una tercera autoridad de confianza (llamada *distribuidor*) o conjuntamente por los n participantes. La entrada es un parámetro de seguridad k. Las salidas son unos parámetros públicos utilizados en todo el esquema, junto con cierta información privada sh_j para cada participante j, que va a ser usada más tarde en el proceso de verificación distribuida.
2) *Gen_Cla*: Este protocolo tiene como salida un par de claves (sk_i, pk_i) para cada usuario i, donde sk_i es la clave privada y pk_i es la correspondiente clave pública.
3) *Firm*: Este algoritmo es ejecutado por el firmante i; toma como entrada un mensaje m y la clave privada sk_i, y da como salida una firma $\theta(m)$ de una función hash del mensaje. Escribiremos $Sign(m, sk_i) = \theta(m)$.
4) *Ver_Dist*: Dado $A \in \Gamma^{(i)}$ un subconjunto autorizado de verificadores para el firmante i, este protocolo toma como entrada un mensaje m, una firma $\theta(m)$, la clave pública pk_i del firmante i y los fragmentos sh_j de los participantes $j \in A$. La salida será 1 si $\theta(m)$ es una firma válida de m y 0 en el caso contrario. Escribiremos $Ver(m, \theta, pk_i, \{sh_j\}_{j \in A}) = 1$ ó 0

Hay que remarcar que el primer y segundo protocolos se ejecutan sólo una vez. Los otros dos protocolos, i.e. el proceso de firma y de verificación, se ejecutan tantas veces como los participantes quieran firmar o verificar.

Si consideramos la familia particular de estructuras de acceso en que $\Gamma_0^{(i)} = \{\{j\} | j \in \mathcal{P} - \{i\}\}$ para todo participante $i \in \mathcal{P}$, obtenemos como resultado un esquema de firma digital casi estándar, definido en un conjunto de participantes en el que todo el mundo puede firmar y verificar. La única diferencia con un esquema de firma estándar es que nadie fuera del sistema puede verificar.

III. Propiedades requeridas

En esta sección definiremos formalmente las tres propiedades que debe cumplir un esquema de firma con verificación distribuida. Antes de eso, explicamos dichas propiedades de manera informal, intuitiva.

1) *Corrección*: siempre que la firma sea generada tras haber seguido los protocolos correctamente, el resultado de la verificación será siempre 1, si el subconjunto de verificadores que participa es autorizado.
2) *Infalsificabilidad*: consideramos la definición de infalsificabilidad más fuerte posible, i.e. infalsificabilidad existencial contra ataques de mensaje escogido [7]. Dicha propiedad requiere que cualquier atacante debe tener probabilidad despreciable de falsificar una firma válida de un usuario que no ha corrompido, incluso si el atacante puede obtener previamente otros pares (mensaje,firma) válidos, para mensajes que él escoge adaptativamente.
3) *Privacidad*: un subconjunto que no sea autorizado para un cierto firmante i no debe obtener ninguna información sobre si una firma computada por i es válida o no. Formalizaremos esta propiedad con un juego de indistinguibilidad.

A. Corrección

Esta propiedad asegura que cuando una firma es generada por un participante siguiendo los protocolos de una manera correcta, el resultado de la verificación es siempre 1, si el subconjunto de verificadores es autorizado respecto a ese firmante. Formalmente:

$$Ver(m, \theta(m), pk_i, \{sh_j\}_{j \in A}) = 1.$$

para todo participante $i \in \mathcal{P}$ y todo subconjunto autorizado $A \in \Gamma^{(i)}$.

B. Infalsificabilidad

El esquema debe ser seguro incluso en presencia de un atacante $\mathcal{F}$ que corrompe algunos participantes del sistema, con la excepción del firmante. Esta propiedad, conocida como *infalsificabilidad existencial contra ataques de mensaje escogido*, se formaliza con el siguiente juego, en el que un retador externo reta a un atacante $\mathcal{F}$ para que intente falsificar una firma válida.

1) El atacante $\mathcal{F}$ recibe como entrada la descripción de un conjunto de participantes $\mathcal{P}$ y, para cada $i \in \mathcal{P}$, la descripción de una estructura de acceso $\Gamma^{(i)} \subset 2^{\mathcal{P}-\{i\}}$.
2) $\mathcal{F}$ escoge un participante i para ser atacado. $\mathcal{F}$ puede corromper al resto de participantes.
3) $\mathcal{F}$ obtiene del retador externo toda la información que se difunde en una ejecución global de los protocolos *Ini* y *Gen_Cla*, i.e. todas las claves públicas, así como la información secreta de los participantes corrompidos (que pueden ser todos menos i).
4) Si la seguridad se considera en el modelo del oráculo aleatorio [1], $\mathcal{F}$ puede hacer Q peticiones al oráculo que modela el comportamiento de una cierta función de hash.
5) $\mathcal{F}$ puede escoger, de manera adaptativa, Q_s mensajes m_ℓ. El protocolo *Sign* es ejecutado por el retador externo, obteniendo $Sign(m_\ell, sk_i) = \theta(m_\ell)$. La firma resultante, $\theta(m_\ell)$, se envía al atacante $\mathcal{F}$.

En un cierto momento, $\mathcal{F}$ publica un par $(m, \theta(m))$. El atacante $\mathcal{F}$ tiene éxito en este juego, si $(m, \theta) \neq (m_\ell, \theta(m_\ell))$, para toda firma obtenida durante el ataque, y además $Ver(m, \theta(m), pk_i, \{sh_j\}_{j \in A}) = 1$, para todo subconjunto $A \in \Gamma^{(i)}$.

Definición 1: Un esquema de firma con verificación distribuida es $\left(\mathcal{P}, \{\Gamma^{(i)}\}_{i \in \mathcal{P}}, T, \varepsilon, Q_s\right)$-infalsificable si no existe ningún atacante $\mathcal{F}$, con tiempo de ejecución como máximo T, que tenga probabilidad de éxito mayor o igual que ε, después de haber pedido como mucho Q_s firmas de manera adaptativa.

C. Privacidad

Antes de definir formalmente la propiedad de privacidad para un esquema de firma con verificación distribuida, debemos introducir la noción de *estructuras de adversario*, denotadas por $\{\Lambda^{(i)}\}_{i \in \mathcal{P}}$. Una estructura de adversario $\Lambda^{(i)}$, para un cierto usuario i es una familia de subconjuntos, que debe cumplir $\Lambda^{(i)} \subset 2^{\mathcal{P}-\{i\}} - \Gamma^{(i)}$. De esta manera, se garantiza que un subconjunto que esté en en $\Lambda^{(i)}$ no puede estar en $\Gamma^{(i)}$. En otras palabras, un adversario no podrá corromper un subconjunto de usuarios autorizado a verificar una firma.

$\Lambda^{(i)}$ debe ser monótona decreciente: si el esquema permanece seguro contra un adversario que corrompe los usuarios de un subconjunto $B_1 \in \Lambda^{(i)}$, entonces el esquema debe permanecer seguro también si un adversario corrompe los usuarios de un subconjunto $B_2 \subset B_1$. Esto implica que la estructura de adversario $\Lambda^{(i)}$ queda determinada por su base $\Lambda_0^{(i)} = \{B \in \Lambda^{(i)} \mid B \cup \{j\} \notin \Lambda^{(i)}, \text{ para todo } j \notin B\}$.

Intuitivamente, en una firma digital con verificación distribuida se requiere que un subconjunto de usuarios que esté en la estructura de adversario $\Lambda^{(i)}$ no pueda obtener ninguna información sobre la (in)validez de las firmas calculadas por el usuario i. Para formalizar exactamente qué quiere decir 'no obtener ninguna información', se adapta el concepto de seguridad semántica (inicialmente introducido para esquemas de cifrado [6]). De manera informal, dado un mensaje escogido por un atacante, y dos firmas, de las cuales sólo una es válida para ese mensaje, el adversario no debe ser capaz de distinguir entre las dos firmas con probabilidad significativamente mayor

que $1/2$ (respuesta aleatoria). Esta propiedad debe cumplirse incluso si el adversario ha corrompido algún subconjunto de participantes, siempre dentro de la estructura de adversario.

Para formalizar esta idea intuitiva, detallamos aquí el juego que un tal adversario $\mathcal{A}$ intenta ganar a un retador externo.

1) El protocolo *Init* es ejecutado por el retador, y $\mathcal{A}$ recibe la información pública del sistema. Dicha información incluye el conjunto de participantes $\mathcal{P}$, las estructuras de acceso $\{\Gamma^{(i)}\}_{i\in\mathcal{P}}$, y las estructuras de adversario $\{\Lambda^{(i)}\}_{i\in\mathcal{P}}$.
2) El atacante $\mathcal{A}$ escoge un participante i, y un subconjunto no autorizado de verificadores, $B \in \Lambda^{(i)}$, para corromper. Como resultado, $\mathcal{A}$ obtiene $\{sh_j\}_{j\in B}$.
3) El protocolo *Gen_Cla* es ejecutado para todos los usuarios. El atacante $\mathcal{A}$ obtiene todas las claves públicas, y las claves privadas $\{sk_j\}_{j\in B}$ de los usuarios corruptos.
4) $\mathcal{A}$ puede pedir firmas válidas del firmante i, para mensajes m_ℓ de su elección (adaptativa).
5) $\mathcal{A}$ escoge, de manera adaptativa, Q_s pares diferentes $(m_\ell, \theta(m_\ell))$, junto con un subconjunto de verificadores autorizado, $A_\ell \in \Gamma^{(i)}$. El retador ejecuta el protocolo $Ver(m_\ell, \theta(m_\ell), pk_i, \{sh_j\}_{j\in A_\ell})$. Como resultado, $\mathcal{A}$ debe recibir toda la información pública que se difunde en dicha ejecución del protocolo de verificación, junto con la información privada que los usuarios corruptos obtienen en esa ejecución.
6) $\mathcal{A}$ escoge y publica un mensaje m.
7) El retador escoge un bit aleatorio $b \in \{0,1\}$, y un mensaje aleatorio $m' \neq m$. Si $b = 1$, el retador ejecuta $Sign(m, sk_i) = \theta^*$. Si $b = 0$, el retador ejecuta $Sign(m', sk_i) = \theta^*$. En cualquier caso, la firma resultante θ^* es enviada al adversario $\mathcal{A}$.
8) $\mathcal{A}$ devuelve un bit $b' \in \{0,1\}$.

Decimos que $\mathcal{A}$ gana el juego si $b' = b$. La *ventaja* de un tal adversario $\mathcal{A}$ se define como $\Pr[b' = b] - 1/2$.

Definición 2: Un esquema de firma con verificación distribuida tiene $(\mathcal{P}, \Gamma, \Lambda, T, \varepsilon, Q_s)$-privacidad fuerte si cualquier adversario $\mathcal{A}$ con tiempo de ejecución como mucho T y con acceso a Q_s ejecuciones del protocolo de verificación, tiene una ventaja que es como mucho ε.

Esta definición de privacidad es la más estricta posible. Se podrían considerar otras nociones de privacidad más débiles, por ejemplo si el adversario $\mathcal{A}$ no tiene permiso para pedir la ejecución del protocolo de verificación para pares (mensaje,firma) de su elección. En otras palabras, el juego que intenta ganar el adversario sería el mismo que en el caso de la privacidad fuerte, con la única diferencia que el paso 5) es eliminado. Si un esquema resiste este tipo de ataques, es decir si un adversario $\mathcal{A}$, que corre en tiempo polinómico y que no puede ejecutar el paso 5), tiene ventaja despreciable, entonces decimos que el esquema disfruta la propiedad de *privacidad débil*.

Formalmente, decimos que una función f es *despreciable* (o *negligible*, en inglés) en k si existe un polinomio $p(\cdot)$ y un valor entero positivo k_0 tal que $f(k) \leq 1/p(k)$ para todo $k \geq k_0$. Usualmente, se escribe $f(k) = \text{negl}(k)$ para las funciones f despreciables en k.

En el esquema que proponemos a continuación, no hemos sido capaces aún de demostrar formalmente la privacidad fuerte (problema abierto), pero sí que incluimos la demostración de la privacidad débil, en la Sección V-C.

IV. El nuevo esquema

Proponemos en esta sección un esquema específico de firma con verificación distribuida. El objetivo es que el esquema sea eficiente, en particular más eficiente que la solución que consistiría en combinar un esquema de firma digital con un esquema de cifrado con descifrado distribuido.

Sea $\mathcal{P} = \{1, \ldots, n\}$ el conjunto de usuarios, y sea $\Gamma^{(i)}$ la estructura de acceso para cada participante i de $\mathcal{P}$. Consideramos un esquema *lineal* de compartición de secretos [16] para cada estructura $\Gamma^{(i)}$. Por simplicidad en la descripción del esquema, supondremos el caso ideal, más eficiente, en el que dichos esquemas para compartir secretos son de *espacio vectorial* [2]: para cada $i \in \mathcal{P}$, existen un entero positivo t_i y una aplicación $\psi_i : \mathcal{P} \to (\mathbb{Z}_q)^{t_i}$, donde q es un número primo grande, tal que $A \in \Gamma^{(i)}$ si y sólo si $\psi_i(i) \in < \{\psi_i(j)\}_{j\in A} >$. Es decir, si el vector $\psi_i(i)$ es combinación lineal de los vectores asignados a los usuarios en A. En esta situación, i puede repartir un secreto entre el resto de participantes de $\mathcal{P}$, simplemente cogiendo un vector aleatorio $v_i \in (\mathbb{Z}_q)^{t_i}$ y calculando $s_j^{(i)} = v_i \cdot \psi_i(j) \in \mathbb{Z}_q$, para todo $j \in \mathcal{P}$. El secreto será $s_i^{(i)}$, y el fragmento de cada usuario $j \neq i$ será $s_j^{(i)}$. Si los usuarios en un conjunto autorizado $A \in \Gamma^{(i)}$ quisieran recuperar el secreto a partir de sus fragmentos, deberían primero calcular los coeficientes $\{\lambda_j^A\}_{j\in A}$ tales que $\psi_i(i) = \sum_{j\in A} \lambda_j^A \psi_i(j)$. Después, simplemente deberían utilizar esos coeficientes para calcular $\sum_{j\in A} \lambda_j^A s_j^{(i)} = s_i^{(i)}$.

Detallamos a continuación los protocolos que componen nuestro esquema de firma con verificación distribuida. Está inspirado en el esquema de firma de Schnorr [14].

1) ***Ini.*** Dado un parámetro de seguridad k, se escogen dos números primos p y q de forma que $q|(p-1)$ y $q \geq 2^k$ (i.e. q tiene al menos k bits). También se tiene que escoger un elemento $g \in \mathbb{Z}_p^*$ con orden q. Para cada participante $i \in \{1, \ldots, n\}$ se hacen públicas la estructura de acceso $\Gamma^{(i)}$ y las aplicaciones $\psi_i : \mathcal{P} \to (\mathbb{Z}_q)^{t_i}$, que realizan $\Gamma^{(i)}$ como estructura de espacio vectorial. Posteriormente se escoge y publica una función de hash $H : \{0,1\}^* \times \mathbb{Z}_p \to \mathbb{Z}_q$. En la demostración de seguridad, supondremos que esta función H se comporta como un oráculo aleatorio [1]. Finalmente, cada usuario $i \in \mathcal{P}$ ejecuta el siguiente protocolo:
   a) Escoge un vector aleatorio $v_i \in (\mathbb{Z}_q)^{t_i}$, y calcula $s_j^{(i)} = v_i \cdot \psi_i(j) \in \mathbb{Z}_q$, para todo $j \in \mathcal{P}$.
   b) Guarda $s_i^{(i)}$ en secreto, y utiliza un canal privado para enviar $s_j^{(i)}$ a cada participante $j \in \mathcal{P}$, $j \neq i$.

   Al final de este protocolo, cada usuario $j \in \mathcal{P}$ tiene un vector secreto de fragmentos $sh_j = (s_j^{(1)}, s_j^{(2)}, \ldots, s_j^{(n)})$.

2) ***Gen_Cla.*** Cada participante i escoge como clave secreta un elemento aleatorio $x_i \in \mathbb{Z}_q^*$. La clave pública correspondiente es $y_i = g^{x_i} \bmod p$.
3) ***Firm.*** Si el participante i quiere firmar un mensaje $m \in \{0,1\}^*$, escoge un valor aleatorio $\alpha \in \mathbb{Z}_q$ y calcula $R = g^\alpha \bmod p$ y $\sigma = \alpha + a_i H(m,R) \bmod q$, donde $a_i = (\alpha + x_i) \cdot s_i^{(i)} \bmod q$. La firma final se define como el par $\theta(m) = (R, \sigma)$.
4) ***Ver_Dist.*** Si los participantes de un subconjunto $A \in \Gamma_0^{(i)}$ quieren verificar la firma $\theta(m) = (R,\sigma)$ del mensaje m, ejecutan el siguiente protocolo.
   a) Cada verificador $j \in A$ calcula $z_j = (Ry_i)^{s_j^{(i)} H(m,R)} \bmod p$ y la introduce como entrada en un algoritmo 'combinador'[1].
   b) El algoritmo 'combinador', tras recibir todos los valores $\{z_j\}_{j\in A}$, encuentra los valores $\{\lambda_j^A\}$ que verifican $\psi_i(i) = \sum_{j\in A} \lambda_j^A \psi_i(j)$.
   c) El algoritmo 'combinador' calcula y publica el producto $Z = \prod_{j\in A} z_j^{\lambda_j^A} \bmod p$.
   d) Para verificar la validez de la firma $\theta(m) = (R,\sigma)$, se comprueba si $g^\sigma = R \cdot Z \bmod p$.

Hay que remarcar que una coalición de participantes $A \in \Gamma_0^{(i)}$ podría obtener el secreto $s_i^{(i)}$ de i a partir de sus propios fragmentos $s_j^{(i)}$. Sin embargo, esto no significa que puedan firmar en nombre de i, ya que de todos modos no tienen acceso a la clave privada x_i, que es necesaria para obtener $a_i = (\alpha + x_i) \cdot s_i^{(i)}$ y firmar. Esta propiedad de infalsificabilidad se demostrará de manera formal en la Sección V-B.

La longitud de las firmas no crece con el número de verificadores. Por tanto, el esquema no tiene inconveniente si el número de verificadores es muy elevado. Además, ejecutar el protocolo *Sign* tiene un coste computacional equivalente a sólo una exponenciación modular en $< g >$ (el resto de operaciones modulares, módulo q, pueden realizarse de manera muy eficiente).

El paso en el que los participantes en A usan un algoritmo 'combinador' de tipo caja negra para calcular el producto Z podría sustituirse por un protocolo de computación multiparte $(|A|,|A|)-$umbral.

V. Análisis del esquema

En este apartado demostramos que el esquema de firma con verificación distribuida propuesto en la sección anterior satisface las tres propiedades requeridas.

[1]Suponemos la existencia de un algoritmo de tipo 'caja negra' (*black-box*) donde cada usuario $j \in A \in \Gamma_0^{(i)}$ pone su valor z_j. Nadie puede obtener las demás entradas, pero sí el resultado que procede de la caja negra

A. Corrección

Si todos los protocolos del esquema se ejecutan correctamente, tendremos que:

$$\begin{aligned} R \cdot Z = R \cdot \prod_{j\in A} z_j^{\lambda_j^A} &= g^\alpha \cdot (Ry_i)^{H(m,R)\sum_{j\in A}\lambda_j^A s_j^{(i)}} \\ &= g^\alpha \cdot g^{(\alpha+x_i)s_i^{(i)}H(m,R)} \\ &= g^{\alpha + a_i H(m,R)} \\ &= g^\sigma \bmod p \end{aligned}$$

y por tanto una firma generada de manera correcta siempre da resultado 1 en la verificación.

B. Infalsificabilidad

Antes de proceder con la demostración de la infalsificabilidad, recordamos dos resultados que serán utilizados en la misma. El primero es una modificación del *Forking Lemma*, introducido por D. Pointcheval y J. Stern en [12]. El segundo es un resultado conocido de probabilidad. Para el primer resultado, hay que recordar la noción de un esquema de firma *genérico*: las firmas tienen la forma (m,R,h,σ), donde R es un valor aleatorio escogido dentro de un conjunto muy grande (de tamaño exponencial en el parámetro de seguridad), y $h = H(m,R)$ para una función de hash H.

Lema 1: (*Forking Lemma* modificado) Consideremos un esquema de firma digital genérico con parámetro de seguridad $k \geq 5$. Sea $\mathcal{B}$ una máquina de Turing en tiempo polinómico probabilístico cuyas entradas consisten sólo en elementos públicos. Denotamos por Q el número de peticiones que $\mathcal{B}$ puede preguntar al oráculo aleatorio. Supongamos que, en tiempo como máximo T y con probabilidad al menos ε, $\mathcal{B}$ produce una firma válida (m,R,h,σ). Entonces existe otra máquina $\mathcal{B}'$ que tiene control sobre $\mathcal{B}$ y que produce, en un tiempo esperado $T' \leq 2T$ y con probabilidad $\varepsilon' \geq \frac{\varepsilon^2}{19Q}$, dos firmas válidas (m,R,h,σ) y (m,R',h',σ') tales que $h \neq h'$.

Hecho 1: La probabilidad de que, al escoger K valores uniforme e independientemente en un conjunto de q elementos, donde $K \leq q$, al menos dos de ellos sean iguales, es menor que $\frac{K^2}{2q}$.

A continuación demostramos que nuestro esquema de firma con verificación distribuida es infalsificable, en el modelo del oráculo aleatorio [1]. La demostración es por reducción, suponiendo que el problema del logaritmo discreto es computacionalmente irresoluble, en grupos de orden primo.

Teorema 1: Supongamos que existe un atacante $\mathcal{F}$ contra la infalsificabilidad de nuestro esquema, que corre en tiempo $\leq T$ y tiene probabilidad de éxito $\geq \varepsilon$, tras hacer Q peticiones al oráculo aleatorio y Q_s peticiones de firma, donde Q y Q_s son cantidades polinómicas en el parámetro de seguridad k. Entonces el problema del logaritmo discreto en $< g >$ se puede resolver en tiempo $T' \leq 2T + (2n + 4Q_s)T_{exp}$ y con probabilidad $\varepsilon' \geq \frac{\varepsilon^2}{19Q} - \text{negl}(k)$, donde n es el número de participantes en $\mathcal{P}$ y T_{exp} es el tiempo requerido para una exponenciación modular en $< g >$.

Proof: Sea (p,q,g,y) una instancia del problema del logaritmo discreto en el subgrupo $< g > \subset \mathbb{Z}_p^*$ de orden q, donde q es un primo que divide $p-1$ y $y \in < g >$.

Utilizaremos el hipotético falsificador $\mathcal{F}$ contra el esquema de firma con verificación distribuida para resolver esta instancia del problema del logaritmo discreto; es decir, para encontrar un valor x tal que $y = g^x \bmod p$.

Puesto que estamos suponiendo que la función de hash H se comporta como un oráculo aleatorio, el atacante $\mathcal{F}$ tendrá permiso para hacer Q peticiones al oráculo aleatorio que modela el comportamiento de H. La idea es que vamos a construir una máquina de Turing $\mathcal{B}$ que simulará el entorno de $\mathcal{F}$, respondiendo a todas sus peticiones con información indistinguible de la que $\mathcal{F}$ obtendría en una ejecución real de los protocolos. Después, aplicaremos el Lema 1 a esta máquina $\mathcal{B}$.

1) Iniciamos $\mathcal{B}$ dándole como entrada la instancia (p, q, g, y) del problema del logaritmo discreto. El primer paso de $\mathcal{B}$ es darle a $\mathcal{F}$ la información general del sistema: el conjunto de participantes $\mathcal{P}$, las diferentes estructuras de acceso $\Gamma^{(i)}$ y las aplicaciones ψ_i que realizan $\Gamma^{(i)}$ como estructura de espacio vectorial, para todo $i \in \mathcal{P}$.
2) $\mathcal{F}$ escoge un usuario i como objetivo de su ataque.
3) $\mathcal{B}$ ejecuta el protocolo *Ini* y envía todos los fragmentos $\{sh_j\}_{j\in\mathcal{P}}$ resultantes a $\mathcal{F}$, donde $sh_j = (s_j^{(1)}, s_j^{(2)}, \ldots, s_j^{(n)})$. Además, escoge pares de claves (sk_j, pk_j) para todo usuario $j \neq i$, donde $sk_j \in \mathbb{Z}_q^*$ se escoge aleatoriamente, y $pk_j = g^{sk_j}$. Para el usuario i, se define $pk_i = y$. Se envía a $\mathcal{F}$ la clave pública pk_i, y el resto de claves $\{(sk_j, pk_j)\}_{j\in\mathcal{P}, j\neq i}$.
4) $\mathcal{F}$ puede hacer Q peticiones al oráculo aleatorio para H. Para contestar estas peticiones, mantenemos una tabla TAB. La primera vez que se hace una petición (m, R), se escoge un valor aleatorio $h \in \mathbb{Z}_q$, se devuelve el valor h a $\mathcal{F}$, y se guarda la relación $H(m, R) = h$ en TAB. Si la misma petición (m, R) se hace en el futuro, buscamos en la tabla y devolvemos a $\mathcal{F}$ el mismo valor h.
5) $\mathcal{F}$ puede pedir firmas válidas, de acuerdo a las claves de i, para Q_s mensajes m_ℓ que él escoge de manera adaptativa. Para responder a cada una de estas peticiones, $\mathcal{B}$ debe hacer lo siguiente.
   a) Escoger aleatoriamente $h_\ell \in \mathbb{Z}_q$ y $\sigma_\ell \in \mathbb{Z}_q$.
   b) Calcular $R_\ell = g^{\frac{\sigma_\ell}{1+s_i^{(i)}h_\ell}} \cdot y^{\frac{-s_i^{(i)}h_\ell}{1+s_i^{(i)}h_\ell}}$. Si se da el caso que $1 + s_i^{(i)}h_\ell = 0$, entonces se repite el paso anterior, escogiendo otro valor de h_ℓ.
   c) 'Falsificar' el oráculo aleatorio para H, imponiendo la relación $H(m_\ell, R_\ell) = h_\ell$. Más tarde, si $\mathcal{F}$ llama al oráculo aleatorio con entrada (m_ℓ, R_ℓ), se le devolverá el valor h_ℓ.
   d) Devolver la firma $(m_\ell, R_\ell, h_\ell, \sigma_\ell)$.

   Es fácil ver que la firma devuelta es consistente, si no existen colisiones a la hora de 'falsificar' el oráculo aleatorio.

Dos clases de colisiones pueden ocurrir, durante la simulación del oráculo aleatorio y las firmas válidas:

- La primera sucede cuando una tupla (m_ℓ, R_ℓ, h_ℓ) se produce en una simulación de una firma válida, pero el par (m_ℓ, R_ℓ) ya ha sido pedido al oráculo aleatorio con anterioridad, obteniendo otro valor $h'_\ell \neq h_\ell$. La probabilidad de una tal colisión es, como mucho, $Q \cdot Q_s \cdot \frac{1}{q}$.
- La segunda colisión sucede si el mismo par (m', R') aparece en dos simulaciones diferentes para obtener firmas válidas. En ese caso, con casi total probabilidad los dos valores h' asignados serán diferentes, y eso producirá una incongruencia. Sin embargo, usando el Hecho 1, tenemos que la probabilidad de que se produzca una colisión de este tipo durante la simulación del atacante $\mathcal{F}$ es como mucho $\frac{Q_s^2}{2q}$.

Resumiendo, la probabilidad total de colisiones es menor que $\frac{(2Q+Q_s)Q_s}{2q} \leq \frac{(2Q+Q_s)Q_s}{2^{k+1}}$. Esta cantidad es óbviamente despreciable en k, dado que estamos suponiendo que Q y Q_s son valores polinómicos en k. Ahora estamos en disposición de calcular la probabilidad $\varepsilon_{\mathcal{B}}$ de que nuestra máquina de Turing $\mathcal{B}$ obtenga una firma válida, suponiendo que $\mathcal{F}$ produce una firma falsificada (m, R, h, σ) con probabilidad de éxito ε, en el experimento real. Puesto que la simulación de $\mathcal{B}$ es perfecta en el caso de que no se produzcan colisiones, podemos escribir:

$$|\varepsilon_{\mathcal{B}} - \varepsilon| =$$

$$|\Pr[\mathcal{B} \text{ tiene éxito (simulado)}] - \Pr[\mathcal{F} \text{ tiene éxito (real) }]| \leq$$

$$\Pr[\mathcal{B} \text{ no sufre colisiones}] \leq \text{negl}(k).$$

Por tanto, tendremos que $\varepsilon_{\mathcal{B}} \geq \varepsilon - \text{negl}(k)$. Además, el tiempo de ejecución $T_{\mathcal{B}}$ que $\mathcal{B}$ necesita para falsificar una firma es $T_{\mathcal{B}} \leq T + nT_{exp} + 2T_{exp}Q_s = T + (n + 2Q_s)T_{exp}$, donde T_{exp} denota el tiempo necesario para una exponenciación modular en $< g >$ (dichas exponenciaciones aparecen a la hora de calcular las claves públicas pk_j y a la hora de simular las firmas válidas que pide $\mathcal{F}$).

Ahora podemos aplicar el Forking Lemma modificado, descrito en el enunciado del Lema 1, a la máquina de Turing $\mathcal{B}$. De esta manera, después de ejecutar dos veces $\mathcal{B}$, obtendremos en tiempo $T' \leq 2T_{\mathcal{B}}$ y con probabilidad $\varepsilon' \geq \frac{\varepsilon_{\mathcal{B}}^2}{19Q}$ dos firmas válidas (m, R, h, σ) y (m, R, h', σ') tales que $h \neq h'$.

Puesto que las dos firmas son válidas, tenemos que

$$\begin{cases} g^{\sigma} = R \cdot (Ry)^{s_i^{(i)}h} \\ g^{\sigma'} = R \cdot (Ry)^{s_i^{(i)}h'} \end{cases} \quad (1)$$

Dividiendo estas dos ecuaciones, obtenemos $g^{\sigma-\sigma'} = (Ry)^{s_i^{(i)}(h-h')}$, que da lugar a $R = g^{\frac{\sigma-\sigma'}{s_i^{(i)}(h-h')}} \cdot y^{-1}$. Si ahora sustituimos en la ecuación (1) el valor de R por esta expresión, obtendremos finalmente $y = g^x$, donde

$$x = \frac{(\sigma - \sigma')(hs_i^{(i)} + 1) - (h - h')\sigma s_i^{(i)}}{(h - h')s_i^{(i)}}.$$

Por tanto, hemos resuelto el problema del logaritmo discreto de y en base g, con tiempo de ejecución $T' \leq 2T_{\mathcal{B}} \leq 2T + (2n + 4Q_s)T_{exp}$ y probabilidad de éxito

$$\varepsilon' \geq \frac{\varepsilon_{\mathcal{B}}^2}{19Q} \geq \frac{\varepsilon^2}{19Q} - \text{negl}(k).$$

■

C. Privacidad

Como hemos mencionado anteriormente, no hemos sido capaces aún de demostrar que nuestro esquema disfruta la propiedad de privacidad fuerte. Este punto queda como un problema abierto que intentaremos resolver en el futuro. Sin embargo, el esquema disfruta la propiedad de privacidad débil. Demostraremos esta propiedad basándonos en el hecho de que un esquema de compartición de secretos de espacio vectorial tiene seguridad *perfecta*. Esto quiere decir que un atacante $\mathcal{D}$ tiene probabilidad exactamente igual a $1/2$ de ganar el siguiente juego:

1) $\mathcal{D}$ escoge la estructura de acceso $\Gamma^{(i)}$ y la aplicación $\psi_i : \mathcal{P} \to (\mathbb{Z}_q)^{t_i}$ que realiza $\Gamma^{(i)}$ como estructura de espacio vectorial. El objetivo de $\mathcal{D}$ será romper la seguridad del esquema de compartición de secretos, para la estructura $\Gamma^{(i)}$, definido por ψ_i.
2) $\mathcal{D}$ escoge un subconjunto no autorizado de usuarios, $B \subset \mathcal{P} - \{i\}$, $B \notin \Gamma^{(i)}$.
3) Un retador escoge un vector aleatorio $v_i \in (\mathbb{Z}_q)^{t_i}$, y calcula $s_j^{(i)} = v_i \cdot \psi_i(j)$, para todo $j \in \mathcal{P}$.
4) El retador envía los fragmentos $\{s_j^{(i)}\}_{j \in B}$ al atacante $\mathcal{D}$.
5) El retador define $s_1^* = s_i^{(i)}$, por un lado. Por otro lado, escoge aleatoriamente $s_0^* \in \mathbb{Z}_q$, tal que $s_1^* \neq s_0^*$.
6) El retador escoge un bit $b \in \{0, 1\}$ al azar, y envía s_b^* al atacante $\mathcal{D}$.
7) Finalmente, $\mathcal{D}$ devuelve un bit b'.

El atacante $\mathcal{D}$ gana si $b' = b$. En un esquema para compartir secretos perfecto, como por ejemplo los de espacio vectorial, la probabilidad de que un tal atacante $\mathcal{D}$ gane este juego es exactamente $1/2$. Normalmente, se suele utilizar la expresión *seguridad incondicional* para referirse a este hecho, puesto que la probabilidad de éxito de $\mathcal{D}$ no depende de sus recursos computacionales: incluso un atacante con recursos ilimitados no puede comprometer la seguridad del esquema.

Teorema 2: Nuestro esquema de firma con verificación distribuida disfruta la propiedad de privacidad débil, de manera incondicional.

Proof: La demostración vuelve a ser por reducción. Suponemos la existencia de un atacante $\mathcal{A}$ contra la privacidad débil de nuestro esquema, que tiene ventaja ε, y construimos a partir de él un atacante $\mathcal{D}$ que puede romper la seguridad de un esquema de compartición de secretos de espacio vectorial con probabilidad de éxito $1/2 + \varepsilon$. Como esto contradice la seguridad de estos esquemas para compartir secretos, llegaremos a la conclusión que un tal atacante $\mathcal{A}$ contra la privacidad débil de nuestro esquema no puede existir, para cualquier valor $\varepsilon > 0$.

Construyamos, pues, el atacante $\mathcal{D}$, que va ejecutando a su vez el hipotético atacante $\mathcal{A}$.

- $\mathcal{D}$ inicia el atacante $\mathcal{A}$, al cual proporciona la descripción de $\mathcal{P}$, de las estructuras de acceso $\{\Gamma^{(j)}\}_{j \in \mathcal{P}}$ y de las estructuras de adversario $\{\Lambda^{(j)}\}_{j \in \mathcal{P}}$, que $\mathcal{D}$ escoge libremente.
- El atacante $\mathcal{A}$ escoge un participante i, y un subconjunto no autorizado de verificadores, $B \in \Lambda^{(i)}$, para corromper. $\mathcal{D}$ escoge $\Gamma^{(i)}$ para su juego, junto con una aplicación $\psi_i : \mathcal{P} \to (\mathbb{Z}_q)^{t_i}$ que realice $\Gamma^{(i)}$. Además, escoge exactamente el mismo subconjunto no autorizado $B \in \Lambda^{(i)}$. El retador de $\mathcal{D}$ envía los fragmentos $\{s_j^{(i)}\}_{j \in B}$ al atacante $\mathcal{D}$, correspondientes a un cierto valor secreto $s_i^{(i)}$. El atacante $\mathcal{D}$ reenvía dichos fragmentos a $\mathcal{A}$.
- En lo referente a las claves de firma (sk_j, pk_j), el atacante $\mathcal{D}$ las genera por su cuenta, y luego envía a $\mathcal{A}$ todas las claves públicas y las claves secretas de los usuarios en B.
- Si $\mathcal{A}$ pide firmas para mensajes de su elección, $\mathcal{D}$ puede usar la clave secreta sk_i para calcular firmas válidas y enviárselas a $\mathcal{A}$.
- En un momento dado, $\mathcal{A}$ escoge y publica un mensaje m. En este momento, $\mathcal{D}$ pide el reto a su retador, obteniendo un valor s_b^*. Ahora $\mathcal{D}$ debe enviar una firma a $\mathcal{A}$, que obtiene de la siguiente manera: escoge un valor aleatorio $\alpha \in \mathbb{Z}_q$ y calcula $R = g^\alpha \bmod p$ y $\sigma = \alpha + a_i H(m, R) \bmod q$, donde $a_i = (\alpha + sk_i) \cdot s_b^* \bmod q$. La firma final $\theta = (R, \sigma)$ es enviada a $\mathcal{A}$.
- $\mathcal{A}$ devuelve un bit b'. El atacante $\mathcal{D}$ contra el esquema de compartición de secretos devuelve exactamente el mismo bit b' como respuesta a su juego.

La probabilidad de éxito de $\mathcal{D}$ es exactamente la misma que la probabilidad de éxito de $\mathcal{A}$. En efecto, si $b = 1$, entonces $s_b^* = s_1^* = s_i^{(i)}$ y la firma $\theta = (R, \sigma)$ es una firma válida del mensaje m. En cambio, si $b = 0$, entonces $s_b^* = s_0^* \neq s_i^{(i)}$ y la firma $\theta = (R, \sigma)$ no es una firma válida de m, sino una firma de cualquier otro mensaje. Por tanto, si $\mathcal{A}$ acierta con probabilidad $1/2 + \varepsilon$ si la firma θ es válida para m o no, entonces $\mathcal{D}$ está acertando con probabilidad $1/2 + \varepsilon$ si el valor s_b^* recibido es igual al secreto real $s_i^{(i)}$ o no. ■

VI. Conclusiones y Trabajo Futuro

En este trabajo hemos considerado un tipo de firma digital en el que el firmante quiere restringir la capacidad de verificación. Para ello, escoge en el momento de la inicialización del sistema un conjunto de verificadores posibles, y una familia (o estructura de acceso) de subconjuntos autorizados a verificar. Dada una firma, sólo se podrá verificar su validez si cooperan los verificadores de un subconjunto autorizado. En caso contrario, no se puede obtener ninguna información acerca de si el firmante ha firmado un mensaje concreto u otro.

Esta primitiva se puede resolver de manera genérica, combinando un esquema de firma estándar y un esquema de cifrado con descifrado distribuido. Nuestro objetivo es encontrar mejores soluciones, en relación al coste computacional de los protocolos. Proponemos en este trabajo un esquema que es más eficiente que la construcción genérica. Como único punto negativo, cabe destacar que no hemos sido capaces de demostrar aún que nuestro esquema disfruta de la propiedad de privacidad fuerte. Este problema abierto queda como parte del trabajo futuro a realizar.

En nuestro planteamiento, consideramos que cada usuario puede ser firmante y verificador. En nuestra solución, hemos optado por simplicidad por la solución más ineficiente en la que cada usuario reparte los fragmentos correspondientes a su estructura de acceso de manera independiente. Como

consecuencia, cada usuario j recibe y tiene que guardar un vector sh_j de n fragmentos secretos, donde n es el número total de usuarios. Otra línea futura de investigación que parece muy interesante consistiría en intentar mejorar este aspecto de la eficiencia del esquema, considerando por ejemplo qué condiciones deben cumplir las diferentes estructuras de acceso $\Gamma^{(i)}$ para que algunos fragmentos se puedan re-utilizar, y se pueda reducir de esta manera el tamaño del vector de fragmentos sh_j.

REFERENCIAS

[1] M. Bellare y P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proceedings of CCS'93*, ACM Press, pp. 62–73 (1993).

[2] E.F. Brickell. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **9**, pp. 105–113 (1989).

[3] D. Chaum y H. Van Antwerpen. Undeniable signatures. *Proceedings Crypto'89*, LNCS **435**, Springer-Verlag, pp. 212–216 (1989).

[4] D. Chaum. Designated confirmer signatures. *Proceedings Eurocrypt'94*, LNCS **950**, Springer-Verlag, pp. 86–89 (1994).

[5] J. Garay y P. MacKenzie. Abuse-free multi-party contract signing. *P*roceedings of DISC'99, LNCS **1693**, pp. 151-165 (1999).

[6] S. Goldwasser y S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, **28**, pp. 270–299 (1984).

[7] S. Goldwasser, S. Micali y R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. of Computing* **17** (2), pp. 281–308 (1988).

[8] M. Jakobsson, K. Sako y R. Impagliazzo. Designated verifier proofs and their applications. *Proceedings Eurocrypt'96*, LNCS **1070**, Springer-Verlag, pp. 142–154 (1996).

[9] F. Laguillaumie y D. Vergnaud. Multi-designated verifiers signatures. *Proceedings of ICICS'04*, LNCS **3269**, Springer-Verlag, pp. 495–507 (2004).

[10] C.H.Lim y P.J.Lee. Directed signatures and applications to threshold cryptosystems. *Workshop on Security Protocols*, LNCS **1189**, Springer-Verlag, pp. 131–138 (1997).

[11] H. Petersen y M. Michels. On signature schemes with threshold verification detecting malicious verifiers. *Security Protocols Workshop*, LNCS **1361**, Springer-Verlag, pp. 67–78 (1997).

[12] D. Pointcheval y J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology* **13** (3), Springer-Verlag, pp. 361–396 (2000).

[13] R. Rivest, A. Shamir y Y. Tauman. How to leak a secret. *Proceedings of Asiacrypt'01*, LNCS **2248**, Springer-Verlag, pp. 552–565 (2001).

[14] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology* **4**, Springer-Verlag, pp. 161–174 (1991).

[15] V. Shoup y R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, vol. **15** (2), Springer-Verlag, pp. 75–96 (2002).

[16] G.J.Simmons, W. Jackson y K. Martin. The geometry of secret sharing schemes. *Bulletin of the ICA*, **1**, pp. 71–88 (1991).